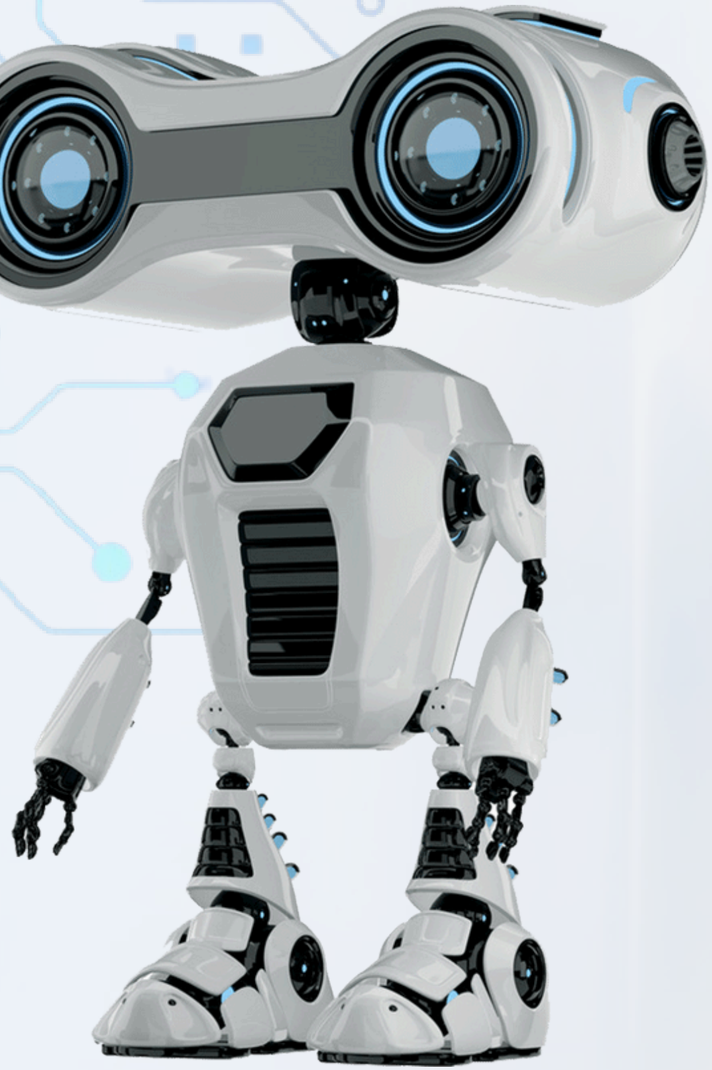
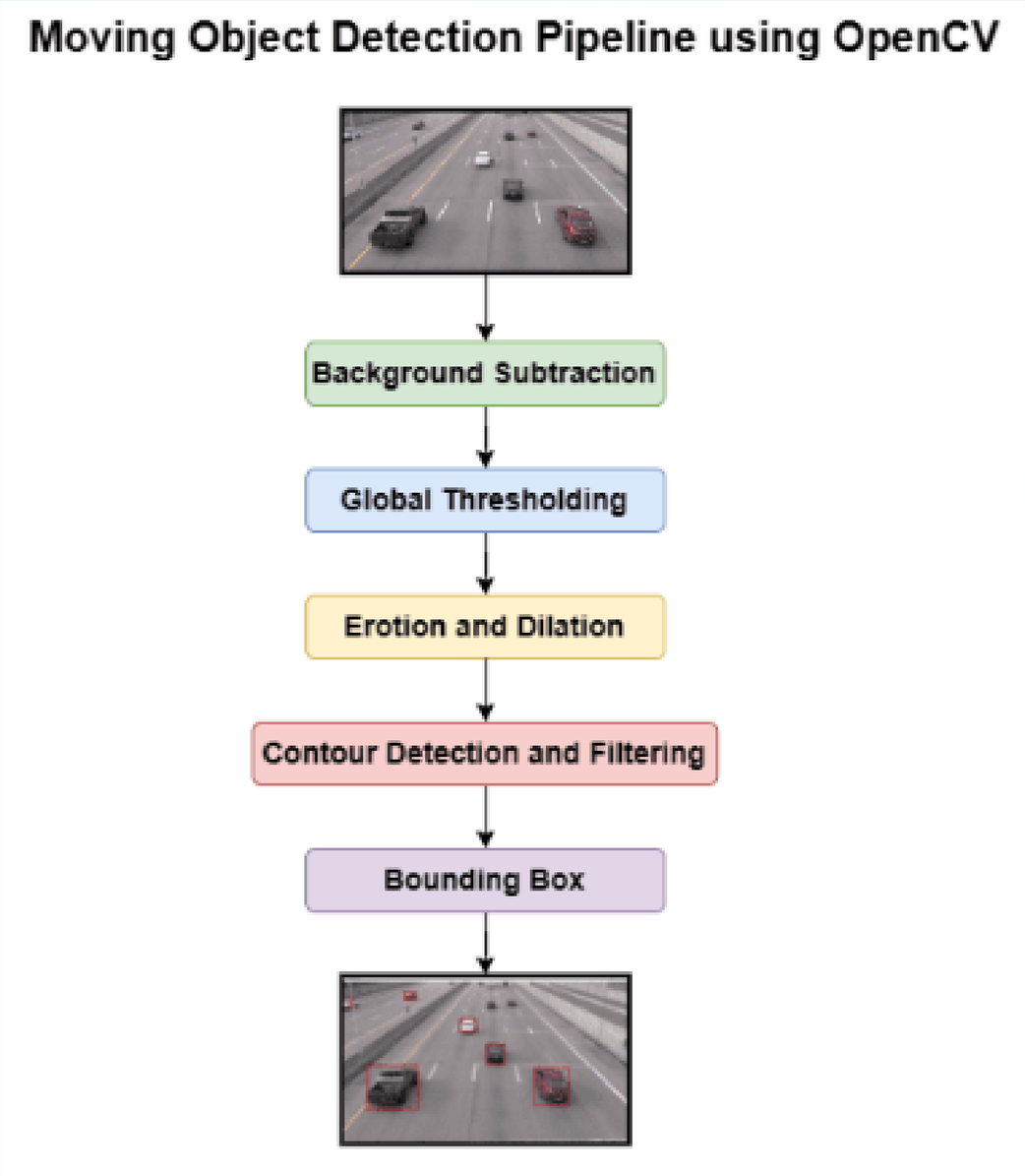
Motion-Based Object Detection in FOG using OpenCV

Real-Time Detection of Moving Objects Without Supervision



Why Motion-Based Object Detection?

- Critical for surveillance, traffic monitoring, and activity tracking.
- Detects moving objects using video streams.
- No need for labeled data or supervised learning.



End-to-End Pipeline

- Frame Capture
- Background Subtraction
- Thresholding & Morphology
- Contour Detection
- Bounding Boxes

What is Background Subtraction?

Background subtraction is a fundamental technique to **isolate moving objects from a static background** in video streams.

Works by

- Comparing each frame with a reference background.
- Identifying significant pixel differences as foreground (moving object).
- Updating the background over time to adapt to scene changes.

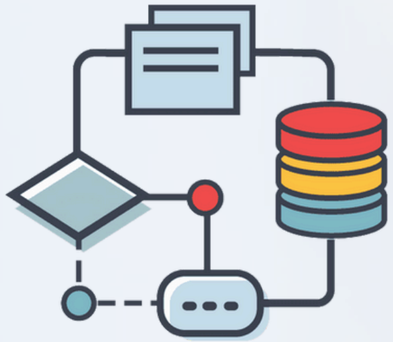
Steps in Background Subtraction

- **Background Initialization** – Learn what the background looks like.
- **Background Update** – Keep adjusting the background model.
- **Foreground Detection** – Spot differences to find moving objects.
- **Post-processing** – Remove noise using morphological operations like **erosion** and **dilation**.

$$|F_t(x,y) - B_t(x,y)| > T$$

Ft: Pixel intensity in current frame at time t
Bt: Background model's predicted pixel value
T:Threshold

Background Subtraction Algorithms



MOG2

- Models each pixel with **K Gaussian distributions**.
- **Continuously adapts** to scene changes.
- **Detects shadows** to avoid false positives (optional and tunable)

KNN

- Classifies foreground based on pixel distance
- Less adaptable than MOG2 but suitable for certain applications

In this project, **MOG2** is chosen. MOG2 is better for **dynamic**, real-world outdoor scenes due to its adaptability and built-in shadow detection

MOG2 Working Principle

🔹 Gaussian Model Initialization

Each pixel is modeled by multiple Gaussians (3–5 per pixel).
Represents pixel value variations over time.

🔹 Continuous Learning (Adaptation)

The model updates dynamically as new frames arrive.
Adapts to lighting changes, moving fog, and scene variations.

Each background pixel value x_N is modelled by a mixture of k Gaussian distributions at time N

$$p(x_N) = \sum_{j=1}^{K} \omega_j \eta(X_N; \theta_j)$$

ω_j is the weight parameter of the k_{th} Gaussian component. $\eta(X;\theta_k)$ is the normal distribution of k_{th} component.

🔹 Foreground Detection

Compares each pixel against the model. Pixels not matching the background model are labeled foreground (moving).

🔹 Mask Cleanup (Post-processing)

Thresholding and morphological operations clean noise and small blobs. Contour detection isolates valid moving objects.

Noise Filtering – Thresholding

Before

1. **Frame Skipping**
- Skips every nth frame to reduce redundancy and improve processing speed.

2. **Resize for Semantic Segmentation**
- Resizes frames to a smaller resolution for faster DeepLabV3 processing.
- DeepLabV3 classifies each pixel into categories like vehicles or roads, and resizing the frame helps speed up the process without losing too much accuracy.

3. **CLAHE (Contrast Limited Adaptive Histogram Equalization)**
- Improves contrast in foggy scenes by applying CLAHE to make vehicles stand out better.
- It works by breaking the image into small parts, adjusting the brightness in each part, and avoiding too much contrast to keep the image clear.

4. **Background Subtraction**
- Uses MOG2 background subtractors to separate the foreground from a static background.

5. **Thresholding**
- Converts the grayscale foreground mask into a binary mask to:
  a. Remove weak motion
  b. Remove shadows
  c. Focus on strong foreground objects (vehicles)

After

Static Camera Simplifies Detection

Assumptions:

- The camera is fixed and stationary.
- The background remains static, detecting only moving objects.

How MOG2/KNN Works:

- Builds a background model from initial frames:
  fg_mask = bg_subtractor.apply(frame)

Foreground mask:

- White (255) = Moving object
- Black (0) = Static background

Why This Simplifies Detection:

- MOG2/KNN works when no background motion exists.
- Clear vehicle detection due to static background.

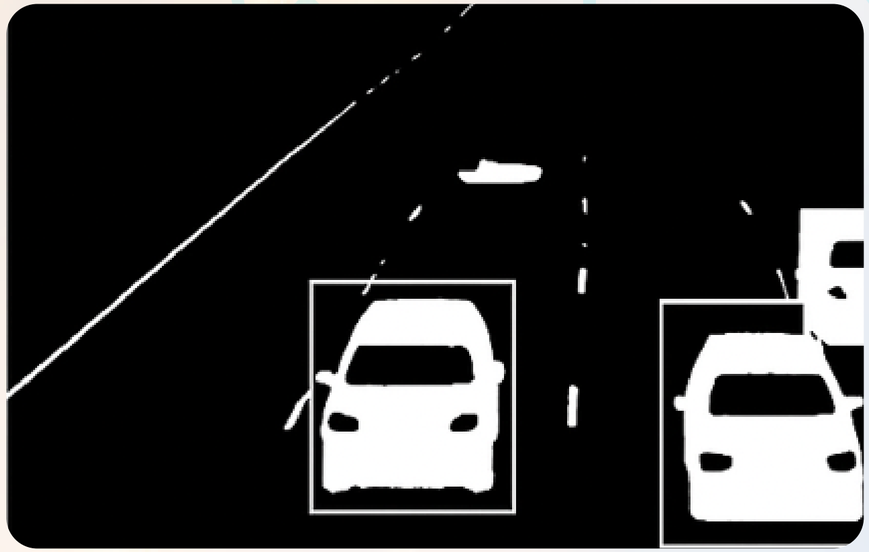
Limitations in Fog:

- Fog and haze may be detected as moving objects (white blobs).
- To differentiate between vehicles and fog, use:
- CLAHE, DeepLabV3, Mask Fusion

Static camera assumption in MOG2:

MOG2_subtractor = cv2.createBackgroundSubtractorMOG2(detectShadows=True, history=500, varThreshold=100)

Relies on a fixed scene for background modeling.



Morphological Operations

Remove Noise with Morphology

- Dilation: Restores shapes by filling gaps.

🔍Why dilation over erosion:

- Joins nearby vehicle parts into one solid blob.
- Fills tiny holes inside blobs (like between windshield and roof).
- Makes the object look more continuous for better contour detection.

```
dilated = cv2.dilate(threshold, cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3)), iterations=1)
```

Contour Filtering

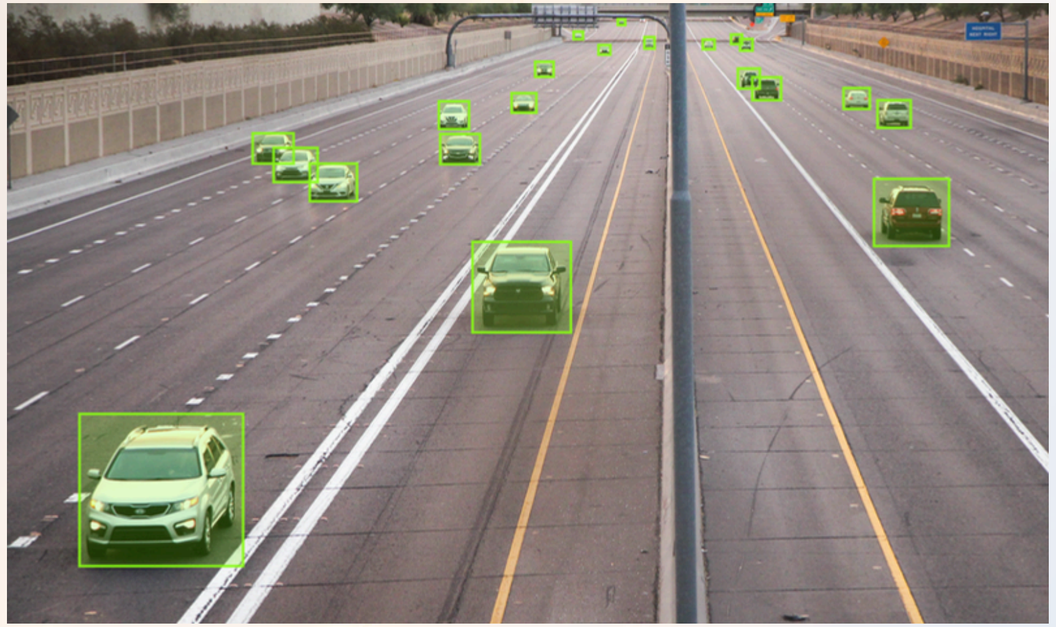
Keep Only Meaningful Shapes

- Filter by area (e.g., > 500 px)
- Remove irrelevant contours
- Focus on prominent objects

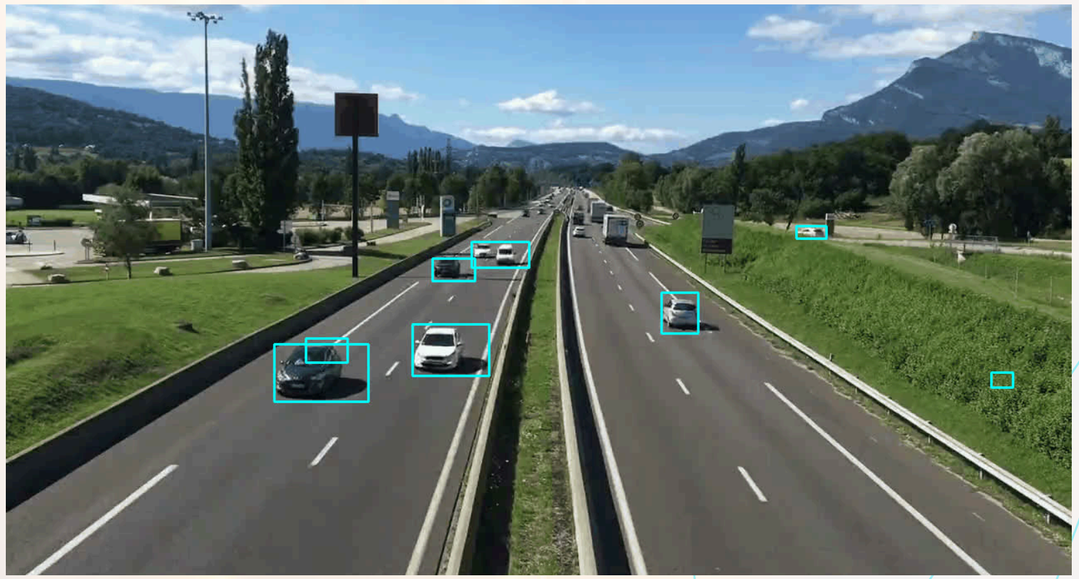
Bounding Boxes

Draw Bounding Boxes Around Objects

- Use cv2.boundingRect()
- Visualize object location with cv2.rectangle()



Moving Object Detection in stable fog free environment



Moving Object Detection in foggy environment

INPUT

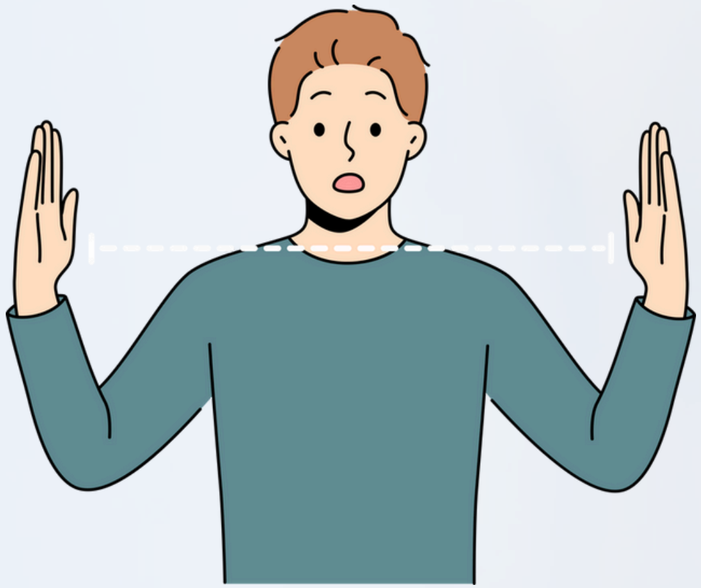
OUTPUT

Result & Analysis

- **Accurate Detection:** Effectively detects moving objects in real-time using MOG2.
- **Optimized Performance:** Frame skipping and resizing improve speed without much accuracy loss.
- **Noise Handling**: Morphological operations and area filtering reduce false detections.
- **Limitations:** Sensitive to lighting changes, fog, and camera movement.
- **Best Use Case:** Works well in static camera setups like traffic and home surveillance.

Performance Parameter	Moving Object Detection	Moving Object Detection in Fog
Total Time to Process (sec)	19.99	913.41
Average FPS	1.00	0.11

Current Limitations



1. Static Camera Assumption
   - The system expects the camera to stay still so if camera moves the system gets confused as background changes.

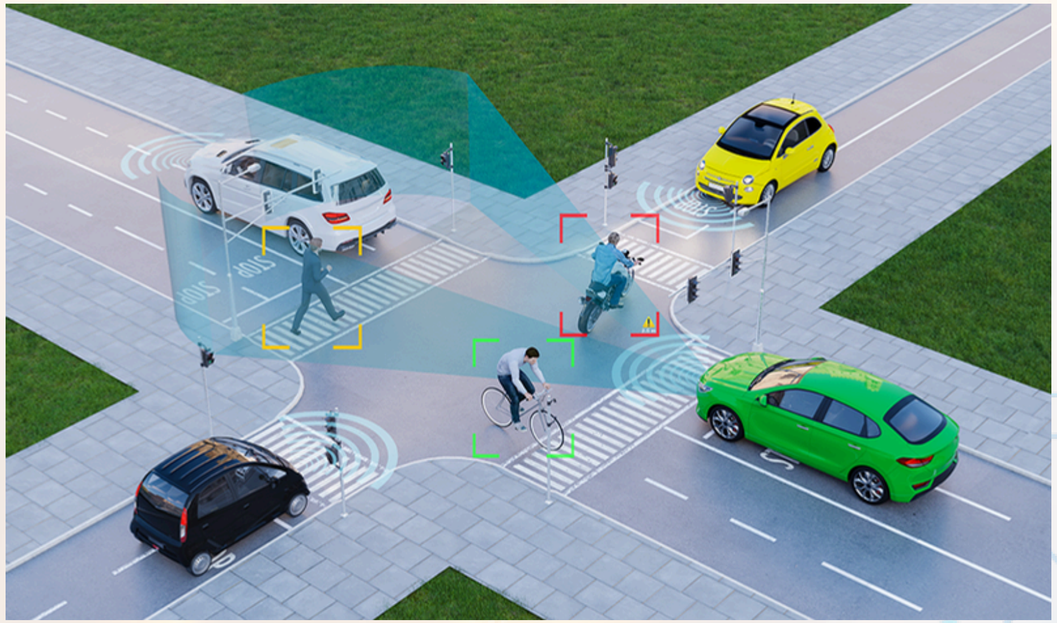
2. Sensitive to Lighting and Weather
   - Sudden lighting changes, like turning on headlights or shadows from clouds, rain, smoke can be detected as motion by mistake.

3. Doesn’t Handle Complex Scenes Well
   - In very crowded environments (like a busy road with lots of people), the system might struggle to distinguish between different objects or may detect too many at once.

Real-World Applications

- Traffic Management

Use Case: Detecting and tracking vehicles on roads.
For example: Automatic number plate recognition (ANPR)



- Home monitoring

Use Case: Monitoring for intruders or activity when the owner is away.
For example: Doorbell cameras (like Ring or Nest)

- Maritime Navigation and Port Monitoring

Use Case: Detecting ships, boats, or other objects in water.
For example: Harbor entry monitoring

Conclusion

- Built a real-time, unsupervised object detection system using OpenCV.
- The system does not need training data, which makes it simple and lightweight.
- This makes it very effective for static camera scenarios like traffic cameras, home security, or parking lots.
- The system is robust in simple environments but could be improved further with AI-based enhancements for more complex scenes.

References

[1] Research Paper
[2] OpenCV Docs
[3] Python OpenCV Tutorials

Thank You!

PRESENTED BY GROUP 11

UCE2022635 - VAISHNAVI SAWANT

UCE2022636 - VEDIKA SAWANT

UCE2022650 - SUSHMA BHANDARY

UCE2022651 - DIKSHA TALATHI

UCE2022670 - ESHITA WAGHODE

UCE2022671 - ANUSHKA YADAV